Connectors in Modelica for Acausal Modeling

Madhusudhan Pandey

October 31, 2025

1 Introduction

Connectors are the philosophical and structural backbone of equation-based modeling languages. They define how energy, information, or material is exchanged between components in a physically meaningful way. In contrast to procedural or block-diagram languages (like Simulink), connectors enable **acausal modeling**[1]—the declaration of relationships rather than **signal flow**.

This report reviews the historical roots, philosophical foundation, core concepts, and modern implementation of connectors in Modelica and Julia's ModelingToolkit.jl.

2 Historical Background

2.1 Modelica Evolution

a) Master Thesis and Simnon: It was autum of 1971 when a compiler technologist, Hilding Elmqvist from the Automatic Control Lab at Lund University of Technology, started his master thesis to develop a simulation program that could take a user input into the models to perfrom some mathematical expressions. The outcome of his master thesis was Simnon (SIMulation of NONlinear systems). Simnon has the following characteristics:

- Early simulations were batch-processed; Simnon allows interactive control through commands.
- It could support continuous, discrete, and connecting systems[2], enabling hybrid simulation (precursor to hybrid DAEs).
- Its development was inspired by the need to model both physical processes (continuous) and digital controllers (discrete).
- It introduces optimization and time-delay subsystems using digital computers.
- The "subsystem" idea anticipates Modelica's object-oriented component-based modeling.
- Integration methods were Adams predictor–corrector and Runge–Kutta methods.
- Simnon was also planned support for stiff systems and *algebraic loops*.

[1] Modeling physical systems has always involved a fundamental question: "*Does nature compute forward in time (cause effect), or does it simply balance relations without an explicit direction?*" This question divides modeling into two philosophical and mathematical paradigms:

a) Causal modeling: models the flow of information (input to output or time convolution).

b) Acausal modeling: models the flow of energy or relations (balance laws or constraints manifold).

Hence, acausal modeling descends from physics first (Energy and relation symmetry), not computation first (Cause–effect determinism). Newton's deterministic laws such as force is a *cause* that determines the acceleration which is an *effect*; hence Newton's definition of force is a causal modeling. The block diagram based modeling in simulink is causal modeling. In contrast, Euler and Lagrange dynamics, Hamiltonian mechanics, Dirac's port based modeling, etc. refer to the acausal modeling. Hilding Elmqvist (1980s) developed Modelica from this energy-exchange philosophy — equations, not instructions.

[2] Continous systems: $\dot{x} = f(x, u, p, t), y = g(x, u, p, t)$, Discrete systems: $x_{k+1} = f(x_k, u_k, p, t_k), y_k = f(x_k, u_k, p, t_k)$ with state updated at sampling instances t_k, and Connecting systems that defines how inputs/outputs are linked. In these equations, x, u, p, and t are state, input, parameter and time, respectively for a dynamical system.

- Events and conditions (like threshold-triggered sampling) were under development — early event-handling concept.
- Simnon introduced its own modeling language, simpler than Fortran, supporting: i) Algebraic assignments (Algol-60–style). and ii) Conditional statements (if–then–else) for modeling *nonlinearities*.
- Simnon has different variable types[3]: INPUT, OUPUT, STATE, DER, NEW, TIME and TSAMP.

"*Simnon was perfect for simulation of sampled data control systems which were becoming very important at that time*" writes Elmqvist, in [2].

b) PhD Thesis: Elmqvist writes in [2] *"However, I was not satisfied with the approach [i.e the Simnon's simulation engine] as a basis for my PhD thesis, since it was based on submodels with inputs and outputs. I therefore wanted to investigate the fundamentals of modeling which my professor Karl Johan Åström supported.*" Karl was the PhD supervisor for Elmqvist, and was suggesting Elmqvist to look into fundamental of physical modeling back in 1976. He suggested Elmqvist to the modeling work from Sture Lindahl (1976) [3]. Lindahl's work were inspired from the modeling work of another scholar at Lund Insitute of Technology, Karl Eklund (1971)[1], who was building linear model of a drum-boiler turbine. This linear modeling work of Karl Eklund was later extended by Sture Lindahl to "*A Non-linear Drum-Boiler Turbine Model*", and it came out as a PhD thesis under the supervision of Karl Johan Åström.

Elmqvist writes "*The report was very systematic, showing how the equations were organized for the drum, the super heaters, the turbines, etc., i.e. the approach was object oriented. The modeling started by stating the relationships as general equations. Then the derivations were shown how to put them into state space form in order to be able to simulate using Simnon. These derivations involved manual symbolic manipulation of individual equations to solve for the unknown, solving linear systems of equations (see excerpt from report below), unrolling Newton´s algorithm a fix number of times to solve nonlinear systems of equations, and differentiating certain equations when there were constraints between differentiated variables (nowadays called index reduction).*" The non-linear modeling work of Lindahl (1976) were well-structured and was simulated in Simnon[4]. The model of the drum-boiler turbine had more than 500 variables and parameters. In Lindahl's 1976 work, all these processes to simulate a non-linear model was done manually; Elmqvist's insight was to make it systematic and automated — one of the core features of Dymola and Modelica. The suggestions provided by *Karl Johan Åström* to look into the work of Lindahl (1976) was insightful for Elmqvist to notice:

a) The model structure was object-oriented (each subsystem was a physical component).

b) But converting it into Simnon's input–output causal ODE form required huge manual symbolic work (derivations, differentiations, Jacobians).

c) Each time a model changed (e.g., new pipe or valve), the entire derivation had to be redone.

These insights were helpful for Elmqvist to understand the need of a new language. He writes in [2]*"Just before Easter 1976, I realized what I needed to do: design a new language allowing general equations, having a class concept for object oriented modeling (I was used to Simula so that was natural) and having a structured feature to describe interaction between submodels (called cut). The language was called Dymola for DYnamic MOdeling LAnguage ".*

c) Dymola: Dymola was a new modeling language that came out of the PhD thesis of Elmqvist. The

[3]Simnon made the derivative an explicit variable — something that could be assigned, checked, and manipulated symbolically. In the 1970s, simulation languages (like SIMSCRIPT, DYNAMO, ACSL) were procedural. Elmqvist's innovation in Simnon was to:

a) Make derivatives explicit (DER variables), b) Introduce sections (OUTPUT, DYNAMICS) that the compiler could analyze separately, c) Allow error checking and consistent symbolic association of derivatives with their corresponding states. This laid the groundwork for: Equation-based modeling (declarative rather than procedural), Symbolic differentiation and integration (which later allowed DAE handling), and Automatic code generation for simulation.

[4]Lindahl's model had constraint equations coupling drum pressure, steam quality, and turbine mass flow. These led to algebraic relations such as:

$f(p_{drum}, h_{steam}, h_{water}) = 0$ and $\dot{m}_{steam} - \dot{m}_{water} = 0$. To simulate, he: a) Differentiated these algebraic relations manually (e.g., total derivative of a constraint) b) Solved the resulting equations for derivatives like $p_{drum}, h_{steam}, h_{water}$etc. c) Substituted back into the dynamic equations, and d) Ensured all derivatives were now explicitly expressed for Simnon. This manual symbolic manipulation i.e. differentiating constraints, eliminating algebraic variables is exactly what modern DAE solvers do automatically through index reduction algorithms (Pantelides algorithm, dummy-derivative method, etc.).

idea of the model language Dymola was to use a concept known as *cut* whcih corresponds to connection mechanism of complex types, and there are facilities to descrbe the connection structure of a system.

2.2 Philosophical Foundation

The philosophy of connectors arises from energy-based system theory and network modeling (e.g., bond graphs, Tellegen's theorem).

At its core a connector represents the interface through which energy or information flows between components, defined by variables that obey physical conservation laws.

References

[1] Karl Eklund. Linear drum boiler-turbine models. *PhD Thesis TFRT-1001*, 1971.

[2] Hilding Elmqvist. Modelica evolution-from my perspective. In *Proceedings of the 10th International Modelica Conference*, volume 2014, 2014.

[3] Sture Lindahl. *A Non-linear Drum-Boiler Turbine Model*. Research Reports TFRT-3132. Department of Automatic Control, Lund Institute of Technology (LTH), 1976.

Table 1: Across and Through Variables in Connector Semantics

Symbol	**Description**	**Physical Example**
Across variable	Equal for connected components	Voltage v, Pressure p, Temperature T
Through variable	Sum to zero at a junction (conserved)	Current i, Mass flow rate $\dot{m}$, Heat flow $\dot{Q}$